INSA de Toulouse, spécialité AEI 4ème année

MATLAB - Quelques commandes

Septembre 2003

1 Variables Matricielles

1.1 Définir une matrice

Pour définir la matrice $A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$

```
>>A=[1 2 3
4 5 6]
```
ou
```
>>A=[1,2,3;4 5 6]
```

1.2 Définir un scalaire

Un scalaire est par définition une matrice de dimensions 1×1 la syntaxe est :

```
>>a=[3.23]
```
ou
```
>>a=3.23
```

Les scalaires sont tous a priori des réels (pas de distinction explicite entre nombres entiers et nombres réels) mais peuvent être complexes :

```
>>z=2.1+44.9i
```
ou
```
>>z=2.1+44.9j
```
ou
```
>>z=2.1+44.9*i
```

Certaines lettres ou chaînes de caractères sont prédéfinies :

`i` et `j` correspondent au nombre imaginaire : $\sqrt{-1}$

`eps` est un réel approchant l'ordre des erreurs de calcul sous MATLAB

`pi`, c'est π

`e` permet d'écrire les puissances de 10 :

`2.34e2` $\rightarrow$ $234 = 2.34 \cdot 10^2$

`Inf` ou `inf` signifie que la variable est infinie : ∞
(`-Inf` est également définie)

`NaN` ou `nan` : "Not a Number"

`[]` : variable vide (i.e. matrice de dimensions nulles).

1.3 Définir un vecteur de réels incrémentés

Le caractère : permet de définir des vecteurs d'entiers successifs :

```
>> 1:5
ans =
     1     2     3     4     5
```

L'incrément par défaut est +1 mais d'autres valeurs sont possibles :

```
>> 5:-1:0
ans =
     5     4     3     2     1     0
>> 0:pi:10
ans =
         0    3.1416    6.2832    9.4248
```

Bien sûr on peut obtenir des résultats rigolos :

```
>> 4:1
ans =
   Empty matrix: 1-by-0
```

1.4 Eléments d'une matrice

Pour obtenir un élément d'une matrice :

```
>>A(2,3)
ans =
     6
```

Pour obtenir une ligne ou une colonne :

```
 >> A(:,3)
 ans =
      3
      6
```

```
>> A(1,:)
ans =
     1     2     3
```

Ces commandes permettent aussi d'extraire des éléments en les réorganisant :

```
>> A([2 1],1:3)
ans =
     4     5     6
     1     2     3
```

1.5 Quelques matrices utiles

`eye(n)` : matrice identité de dimension n

`eye(2)` $\rightarrow$ $\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$

`zeros(m,n)` : matrice de zéros de dimension $m \times n$

`zeros(1,2)` $\rightarrow$ $\begin{bmatrix} 0 & 0 \end{bmatrix}$

`ones(m,n)` : matrice de 1

`ones(2,3)` $\rightarrow$ $\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$

1.6 Opérations usuelles sur les matrices

Addition, soustraction : `A+B` , `A-B`

Multiplications, inversion : `A*B` , `B*A` , `inv(A)`

Conjuguée : `conj(A)` $\rightarrow$ $\overline{A} = [\,\overline{a}_{ij}\,]_{(i=1..n\ ,\ j=1..m)}$

Transpose : `A.'` $\rightarrow$ A^T

Conjuguée transpose : `A'` $\rightarrow$ $A^* = \overline{A}^T$

Exponentielle : `expm(A)` $\rightarrow$ e^A

Puissance : `A^3` $\rightarrow$ A^3

Racine carrée : `sqrtm(A)` $\rightarrow \sqrt{A}$

Concaténation horizontale : `[A,B]` $\rightarrow [\ A \quad B\ ]$

Concaténation verticale : `[A;B]` $\rightarrow \begin{bmatrix} A \\ B \end{bmatrix}$

Valeurs propres : `eig(A)`

rang : `rank(A)`

trace : `trace(A)`

déterminant : `det(A)`

normes : `norm(A)` , `norm(A,1)` , `norm(A,inf)`

polynôme caractéristique : `poly(A)`

Voir aussi `logm`, `svd`...

1.7 Opérations termes à termes

Multiplication : `A.*B` $\rightarrow [\ a_{ij} \cdot b_{ij}\ ]_{(i=1..n\ ,\ j=1..m)}$

Inversion : `1./A` $\rightarrow [\ 1/a_{ij}\ ]_{(i=1..n\ ,\ j=1..m)}$

Division : `B./A` $\rightarrow [\ b_{ij}/a_{ij}\ ]_{(i=1..n\ ,\ j=1..m)}$

Exponentielle : `exp(A)` $\rightarrow [\ e^{a_{ij}}\ ]_{(i=1..n\ ,\ j=1..m)}$

Puissance : `A.^3` $\rightarrow [\ a_{ij}^3\ ]_{(i=1..n\ ,\ j=1..m)}$

Racine carrée : `sqrt(A)` $\rightarrow [\ \sqrt{a_{ij}}\ ]_{(i=1..n\ ,\ j=1..m)}$

Voir aussi `abs`, `log`, `log10`, `real`, `imag`, `round`, `sin`, `acos`, `tanh`...

2 Polynômes

2.1 Ecriture vectorielle

Certaines fonctions utilisent des polynômes qui sont alors définis comme des vecteurs ligne :

$12p^3+6p^2-4p+9$ s'écrit comme le vecteur ligne : `[12 6 -4 9]`

On représente un polynôme par un vecteur ligne où les coefficients sont entrés dans l'ordre décroissant du degré.

2.2 Opérations sur le polynômes

Si -1, -2 et -3 sont les racines d'un polynôme

$$(x+1)(x+2)(x+3)$$

pour calculer la forme développée du polynôme, on fait :

```
>>poly([-1 -2 -3])
ans=
   1 6 11 6
```

Le polynôme obtenu est donc $x^3+6x^2+11x+6$.
Inversement, si on cherche les racines de $x^2-x+2=0$:

```
>>roots([1 -1 2])
ans=
   0.5000 + 1.3229i
   0.5000 - 1.3229i
```

Une autre opération très utile est la décomposition en éléments simples d'une fraction rationnelle. Voici un exemple d'utilisation. Partant de la fraction rationnelle suivante :

$$\frac{p^2+2p+3}{p^4-8p^3+23p^2-28p+12}$$

La décomposition en éléments simples est obtenue avec la commande :

```
>> [R,P,K]= residue([1 2 3], [1 -8 +23 -28 12])
R =
    9.0000
   -6.0000
  -11.0000
   -3.0000

P =
    3.0000
    2.0000
    2.0000
    1.0000

K =
     []
```

Ce qui signifie que la décomposition en éléments simples s'écrit (voir aide de MATLAB pour cette fonction) :

$$\frac{9}{p-3}-\frac{6}{p-2}-\frac{11}{(p-2)^2}-\frac{3}{p-1}+0$$

3 Systèmes Linéaires

Une boite à outils nommée CONTROL SYSTEM TOOLBOX permet de définir des variables MATLAB pour un système linéaire invariant dans le temps (LTI). Cette boite à outils gère de plus les opérations usuelles de l'Automatique.

3.1 Définir une fonction de transfert

3.1.1 Représentation usuelle

Pour définir une fonction de transfert d'un système à temps continu ou bien à temps discret on utilise l'unique fonction `tf` en spécifiant le polynôme du numérateur et le polynôme du dénominateur (voir section précédente sur les polynômes). L'usage est le suivant :

Définir $G_c(p)=\dfrac{2p+1}{p^2+2p+1}$

```
>> gc = tf([2 1], [1 2 1])

Transfer function:
   2 s + 1
-------------
s^2 + 2 s + 1
```

On remarquera que la variable de Laplace a pour notation 's' dans MATLAB (notation anglo-saxonne).

Définir $\dfrac{z-1}{z^2-3z+2}$ à la cadence $T=2s$:

```
>> tf([1 -1], [1 -3 2], 2)

Transfer function~:
    z - 1
-------------
z^2 - 3 z + 2

Sampling time: 2
```

A la donnée d'une représentation sous forme de fonction de transfert MATLAB retourne les vecteurs définissant le numérateur et le dénominateur (et éventuellement la cadence) à l'aide de la commande suivante `[NUM,DEN,T] = tfdata(SYS,'v')` où `SYS` est la variable MATLAB contenant le modèle.

3.1.2 Représentation zéros/pôles/gain

Une autre écriture pour les fonctions de transfert est également acceptée : `zpk`. Elle consiste a factoriser les racines des polynômes numérateur et dénominateur. Par exemple :

```
>> gd = zpk([1 2], [3 -1 4], 10, 1.5)

Zero/pole/gain~:
 10 (z-1) (z-2)
-----------------
(z-3) (z-4) (z+1)

Sampling time: 1.5
```

Il est bien évidement possible de passer d'une représentation à l'autre en utilisant les commandes `tf` et `zpk` comme suit :

```
>> zpk(gc)

Zero/pole/gain:
2 (s+0.5)
---------
 (s+1)^2

>> tf(gd)

Transfer function:
  10 z^2 - 30 z + 20
----------------------
z^3 - 6 z^2 + 5 z + 12

Sampling time: 1.5
```

De même que pour les données de `tf`, pour obtenir les vecteurs comprenant les zéros (racines du numérateur), les pôles (racines du dénominateur) et le gain il suffit d'appliquer la commande `[Z,P,K,T] = zpkdata(SYS,'v')`.

3.2 Définir une représentation d'état

Pour définir un modèle dans l'espace d'état d'un système à temps continu ou bien à temps discret on utilise l'unique fonction `ss` en spécifiant les matrices A (dynamiques), B (entrées), C (mesures) et D (transfert direct). L'usage est le suivant :

Définir F_c :
$$\begin{cases} \dot{x} = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} x + \begin{bmatrix} 5 \\ 6 \end{bmatrix} \\ y = \begin{bmatrix} 7 & 8 \end{bmatrix} x + 9u \end{cases}$$

```
>> fc = ss([1 2; 3 4], [5; 6], [7 8], 9)
a =
       x1  x2
   x1   1   2
   x2   3   4
b =
       u1
   x1   5
   x2   6
c =
       x1  x2
   y1   7   8
d =
       u1
   y1   9
Continuous-time model.
```

La notation est équivalente pour les systèmes à temps discret, il faut spécifier la cadence T en plus des quatre matrices :

```
>> ss(A, B, C, D, T)
```

De même que pour les données de `tf` et `zpk`, pour obtenir les matrices du modèle il suffit d'appliquer la commande `[A,B,C,D,T] = ssdata(SYS)`.

3.3 Manipulations entre modèles

Il est bien sur possible de passer d'une représentation d'état à une fonction de transfert et réciproquement sans difficulté. Pour cela il suffit d'appliquer la fonction `tf`, `zpk` ou `ss` selon le résultat souhaité. Quelques exemples :

```
>> tf(fc)

Transfer function:
9 s^2 + 38 s - 2
----------------
 s^2 - 5 s - 2

>> ss(gd)

a =
          x1      x2      x3
   x1      3  -1.225  0.7071
   x2      0      -1   1.732
   x3      0       0       4

b =
         u1
   x1     0
   x2     0
   x3 6.325

c =
          x1      x2      x3
   y1  4.472  -2.739   1.581

d =
       u1
   y1   0

Sampling time: 1.5
Discrete-time model.
```

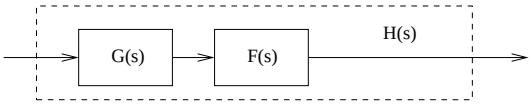
Les représentation d'état n'étant pas uniques il est possible d'opérer des changements de bases classiques. Certaines fonctions sont :

`canon` : donne la forme modale ou bien une forme canonique particulière.

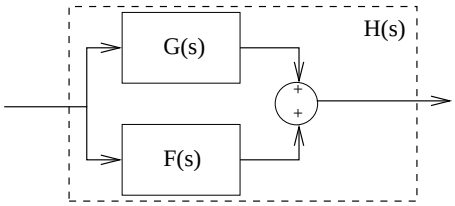
`ss2ss` : applique un changement de base donné.

3.4 Algèbre des systèmes

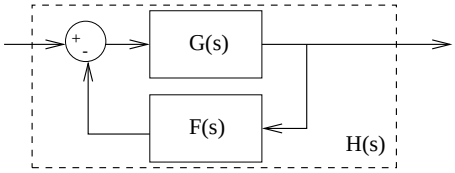
- Systèmes en série `H=F*G`



- Systèmes en parallèle `H=F+G`



- Systèmes bouclés `H=feedback(G,F)`



Attention à ne pas opérer des systèmes de différentes représentations temporelles tels que des systèmes continus avec des systèmes discrets et des systèmes discrets à cadences différentes entre eux.

3.5 Echantillonnage et discrétisation

L'échantillonnage et la discrétisation sont des opérations qui transforment un modèle à temps continu en modèle à temps discret mais ne sont pas du tout équivalentes. La première est une opération physiquement réalisée par un bloqueur d'ordre zéro et un échantillonneur alors que la seconde est purement mathématique.

3.5.1 Echantillonnage

L'opération $G(z) = \mathcal{Z}[B_0(p)G(p)]$ est réalisée par MATLAB comme suit :

```
>> c2d(gc, 1.2, 'zoh')

Transfer function:
    1.06 z - 0.5719
-----------------------
z^2 - 0.6024 z + 0.09072

Sampling time: 1.2
```

3.5.2 Discrétisation

Parmi les méthodes d'approximation mathématique du comportement d'un système continu par un système à temps discret, nous retiendrons la méthode de Tustin. Elle est réalisée avec MATLAB de la façon suivante :

```
>> c2d(gc, 1.2, 'tustin')

Transfer function:
0.6094 z^2 + 0.2812 z - 0.3281
------------------------------
     z^2 - 0.5 z + 0.0625

Sampling time: 1.2
```

3.6 Quelques fonctions utiles

ANALYSE DES PÔLES ET ZÉROS

`pole` : pôles du système

`zero` : zéros du système

`damp` : pôles, les pulsations propres et amortissements associés

`pzmap` : représentation graphique des pôles et zéros dans le plan complexe

RÉPONSES TEMPORELLES

`step` : réponse indicielle

`impulse` : réponse impulsionnelle

`initial` : réponse pour une condition initiale donnée

`lsim` : réponse à une entrée quelconque

`gensig` : génère un signal d'entrée

RÉPONSES FRÉQUENTIELLES

`dcgain` : gain statique

`bandwidth` : fréquence de coupure

`bode` : diagramme de Bode

`margin` : diagramme de Bode + marges de gain/phase

`nyquist` : diagramme de Nyquist

`nichols` : diagramme de Black-Nichols

`grid` : abaques pour le diagramme actif

SYNTHÈSE DE CORRECTEURS

`rlocus` : lieu d'Evans

`ctrb` : matrice de commandabilité

`obsv` : matrice d'observabilité

`place` ou `acker` : placement de pôle

4 Outils Graphiques

Matlab est souvent utilisé pour tracer des courbes et des surfaces car le résultat est très réussi. Les commandes sont assez intuitives et ne méritent pas d'être détaillées ici. Pour connaître une grande variété de commandes tapez `>> help graph2d` pour les courbes en 2D. Et pour les courbes en 3D, vous devinez ce qu'il faut faire.